SWAN

IMPLEMENTATION MANUAL
SWAN Cycle III version 40.91A

by : The SWAN team

mail address : Delft University of Technology
Faculty of Civil Engineering and Geosciences
Environmental Fluid Mechanics Section
P.O. Box 5048
2600 GA Delft
The Netherlands

e-mail : swan-info-citg@tudelft.nl
home page : http://www.swan.tudelft.nl

Copyright (c) 1993-2013 Delft University of Technology.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at http://www.gnu.org/licenses/fdl.html#TOC1.

- Introduction
  - The material

- Use of patch files
- Installation of SWAN on your computer
  - Quick installation
  - Manual installation
    - Modifications in the source code
    - Compiling and linking SWAN source code
  - Building with netCDF support
  - Installation of MPICH2 on Windows XP/Vista/7
  - Make SWAN documentation

- User dependent changes and the file `swaninit`
- Usage of SWAN executable
- Compiling and running unstructured mesh SWAN in parallel
  - Code compilation
  - Running the model

- Testing the system

About this manual

The information about the SWAN package is distributed over four different documents. This User Manual describes the complete input and usage of the SWAN package. The Implementation Manual explains the installation procedure of SWAN on a single- or multi-processor machine with shared or distributed memory. The Programming rules is meant for programmers who want to develop SWAN. The Scientific/Technical documentation discusses the physical and mathematical details and the discretizations that are used in the SWAN program. The mapping of these numerical techniques in SWAN code is also discussed.

In Chapter 2 general description of the model and some instructions concerning the usage of SWAN, the treatment of grids, boundary conditions, etc. is given. It is advised to read this chapter before consulting the rest of the manual. Chapter 3 gives some remarks concerning the input and output files of SWAN. Chapter 4 describes the complete set of commands of the program SWAN.

It is strongly advised that users who are not so experienced in the use of SWAN first read Chapters 2 and 3.

This Manual also contains some appendices. In Appendix A definitions of several output parameters are given. Appendix B outlines the syntax of the command file (or input file). A complete set of all the commands use in SWAN can be found in Appendix C. Appendix D described the format of the files for spectral input and output by SWAN.

General description and instructions for use

Subsections

- Introduction
- Limitations
- Internal scenarios, limiters, shortcomings and coding bugs
- Relation to WAM, WAVEWATCH III and others
- Units and coordinate systems
- Choice of grids, time windows and boundary / initial / first guess conditions
  - Introduction
  - Input grid(s) and time window(s)
  - Computational grids and boundary / initial / first guess conditions
  - Output grids

- Activation of physical processes
- Time and date notation

Introduction

The purpose of this chapter is to give some general advice in choosing the basic input for SWAN computations.

SWAN is a third-generation wave model for obtaining realistic estimates of wave parameters in coastal areas, lakes and estuaries from given wind, bottom and current conditions. However, SWAN can be used on any scale relevant for wind-generated surface gravity waves. The model is based on the wave action balance equation with sources and sinks.

An important question addressed is how to choose various grids in SWAN (resolution, orientation, etc.) including nesting. In general, we consider two types of grids: structured and unstructured. Structured grids may be recti-linear and uniform or curvi-linear. They always consist of quadrilaterals in which the number of grid cells that meet each other in an internal grid point is 4. In unstructured grids, this number can be arbitrarily (usually between 4 and 10). For this reason, the level of flexibility with respect to the grid point distribution of unstructured grids is far more optimal compared to structured grids. Unstructured grids may contain triangles or a combination of triangles and quadrilaterals (so-called hybrid grids). In the current version of SWAN, however, only triangular meshes can be employed.

Often, the characteristic spatial scales of the wind waves propagating from deep to shallow waters are very diverse and would required to allow local refinement of the mesh near the coast without incurring overhead associated with grid adaptation at some distance offshore. Traditionally, this can be achieved by employing a nesting approach.

The idea of nesting is to first compute the waves on a coarse grid for a larger region and then on a finer grid for a smaller region. The computation on the fine grid uses boundary conditions that are generated by the computation on the coarse grid. Nesting can be repeated on ever decreasing scales using the same type of coordinates for the coarse computations and the nested computations (Cartesian or spherical). Note that curvi-linear grids can be used for nested computations but the boundaries should always be rectangular.

The use of unstructured grids in SWAN offers a good alternative to nested models not only because of the ease of optimal adaption of mesh resolution but also the modest effort needed to generate grids about complicated geometries, e.g. islands and irregular shorelines. This type of flexible meshes is particularly useful in coastal regions where the water depth varies greatly. As a result, this variable spatial meshing gives the highest resolution where it is most needed. The use of unstructured grids facilitates to resolve the

model area with a relative high accuracy but with a much fewer grid points than with regular grids.

It must be pointed out that the application of SWAN on ocean scales is not recommended from an efficiency point of view. The WAM model and the WAVEWATCH III model, which have been designed specifically for ocean applications, are probably one order of magnitude more efficient than SWAN. SWAN can be run on large scales (much larger than coastal scales) but this option is mainly intended for the transition from ocean scales to coastal scales (transitions where nonstationarity is an issue and spherical coordinates are convenient for nesting).

A general suggestion is: start simple. SWAN helps in this with default options. Furthermore, suggestions are given that should help the user to choose among the many options conditions and in which mode to run SWAN (first-, second- or third-generation mode, stationary or nonstationary and 1D or 2D).

The material

The downloaded file `swan4091.tar.gz` contains the SWAN source code. You can unzip this file either with WinRAR (in case of Windows) or with the command `tar xzf` (in case of UNIX or Linux). The SWAN source code consists of the following files:

main program	: swanmain.ftn
pre-processing routines	: swanpre1.ftn
	swanpre2.ftn
	SwanBndStruc.ftn90
computational routines	: swancom1.ftn
	swancom2.ftn
	swancom3.ftn
	swancom4.ftn
	swancom5.ftn
post-processing routines	: swanout1.ftn
	swanout2.ftn
service routines	: swanser.ftn
	SwanIntgratSpc.ftn90
routines for support parallel MPI runs	: swanparll.ftn
routines for unstructured grids	: SwanReadGrid.ftn90
	SwanReadADCGrid.ftn90
	SwanReadTriangleGrid.ftn90
	SwanReadEasymeshGrid.ftn90
	SwanInitCompGrid.ftn90
	SwanCheckGrid.ftn90
	SwanCreateEdges.ftn90
	SwanGridTopology.ftn90
	SwanGridVert.ftn90
	SwanGridCell.ftn90
	SwanGridFace.ftn90

SwanPrintGridInfo.ftn90
SwanFindPoint.ftn90
SwanPointinMesh.ftn90
SwanBpntlist.ftn90
SwanPrepComp.ftn90
SwanVertlist.ftn90
SwanCompUnstruc.ftn90
SwanDispParm.ftn90
SwanPropvelX.ftn90
SwanSweepSel.ftn90
SwanPropvelS.ftn90
SwanTranspAc.ftn90
SwanTranspX.ftn90
SwanDiffPar.ftn90
SwanGSECorr.ftn90
SwanInterpolatePoint.ftn90
SwanInterpolateAc.ftn90
SwanInterpolateOutput.ftn90
SwanConvAccur.ftn90
SwanConvStopc.ftn90
SwanThreadBounds.ftn90
SwanFindObstacles.ftn90
SwanCrossObstacle.ftn90
SwanComputeForce.ftn90

routines for parallel,
unstructured grids : SwanReadfort18.ftn90
SwanSumOverNodes.ftn90
SwanMinOverNodes.ftn90
SwanMaxOverNodes.ftn90
SwanPunCollect.ftn90

modules and subroutines
for netCDF : nctablemd.ftn90

	agioncmd.ftn90
	swn_outnc.ftn90
couplers	: couple2adcirc.ftn90
routines for installation	: ocpids.ftn
command reading routines	: ocpcre.ftn
miscellaneous routines	: ocpmix.ftn
general modules	: swmod1.ftn
	swmod2.ftn
modules for XNL	: m_constants.ftn90
	m_fileio.ftn90
	serv_xnl4v5.ftn90
	mod_xnl4v5.ftn90
modules for unstructured	
grids	: SwanGriddata.ftn90
	SwanGridobjects.ftn90
	SwanCompdata.ftn90

The source code is written in Fortran 90. Some routines are written in fixed form and depending on your system, the extension may be `for` or `f`. Other routines are written in free form and are indicated by extension `f90`. The conversion from `ftn` or `ftn90` to one of these extensions can be done automatically or manually; see Chapter 3.

You are allow to make changes in the source code of SWAN, but Delft University of Technology will not support modified versions of SWAN. If you ever want your modifications to be implemented in the authorized version of SWAN (the version on the SWAN web page), you need to submit these changes to the SWAN team (*swan-info-citg@tudelft.nl*). The source code is being attended with the following files:

installation procedures	: INSTALL.README
	Makefile
	Makefile.latex
	macros.inc
	which.cmd
	platform.pl

	switch.pl
	adjlfh.pl
run procedures	: SWANRUN.README
	swanrun
	swanrun.bat
machinefile for parallel	
MPI runs	: machinefile
for concatenation of	
multiple hotfiles	: swanhcat.ftn
	hcat.nml
	HottifySWAN.ftn90
edit file	: swan.edt
Matlab scripts for	
unstructured grids	: plotunswan.m
	plotgrid.m
documentations	: swanuse.tex
	swanuse.pdf
	swanimp.tex
	swanimp.pdf
	swantech.tex
	swantech.pdf
	swanpgr.tex
	swanpgr.pdf
	latexfordummies.tex
	latexfordummies.pdf
	conc.lst
	*.eps

The Latex files (*.tex) can be read and written by any editor and can be compiled with LaTeX2e. This compilation can also be done automatically; see Section 3.5.

On the SWAN web page, you also find some test cases with some output files for making a

configuration test of SWAN on your computer. You may compare your results with those in the provided output files.

Use of patch files

Between releases of authorised SWAN versions, it is possible that bug fixes or new features are published on the SWAN web page. These are provided by patch files that can be downloaded from the web site. Typically, a patch can be installed over the top of the existing source code. Patches are indicated by a link to **patchfile**. The names refer to the current version number supplemented with letter codes. The first will be coded 'A' (i.e. 40.91.A), the second will be coded 'B', the third will be coded 'C', etc. The version number in the resulting output files will be updated to 40.91ABC, indicating the implemented patches.

To use a patch file, follow the next instructions:

1. download the file (right-click the file and choose *save link as*)
2. place it in the directory where the source code of SWAN is located
3. execute `patch -p0 < patchfile`

After applying a patch or patches, you need to recompile the SWAN source code.

It is important to download the patch and not cut and paste it from the display of your web browser. The reason for this is that some patches may contain tabs, and most browsers will not preserve the tabs when they display the file. Copying and pasting that text will cause the patch to fail because the tabs would not be found. If you have trouble with patch, you can look at the patch file itself.

Note to UNIX/Linux users: the downloaded patch files are MS-DOS ASCII files and contain carriage return (CR) characters. To convert these files to UNIX format, use the command `dos2unix`. Alternatively, execute `cat 40.91.[A-C] | tr -d '\r' | patch` that apply the patch files 40.91.A to 40.91.C to the SWAN source code at once after which the conversion is carried out.

Note to Windows users: `patch` is a UNIX command. Download the patch program from the SWAN web site, which is appropriate for Windows operating system (XP/Vista/7).

Installation of SWAN on your computer

The portability of the SWAN code between different platforms is guaranteed by the use of standard ANSI FORTRAN 90. Hence, virtually all Fortran compilers can be used for installing SWAN. See also the manual Programming rules.

The SWAN code is parallelized, which enables a considerable reduction in the turn-around time for relatively large CPU-demanding calculations. Two parallelization strategies are available:

- The computational kernel of SWAN contains a number of OpenMP compiler directives, so that users can optionally run SWAN on a dual core PC or multiprocessor systems.
- A message passing modelling is employed based on the Message Passing Interface (MPI) standard that enables communication between independent processors. Hence, users can optionally run SWAN on a Linux cluster.

The material on the SWAN web site provides a `Makefile` and two Perl scripts (`platform.pl` and `switch.pl`) that enables the user to quickly install SWAN on the computer in a proper manner. For this, the following platforms, operating systems and compilers are supported:

platform	**OS**	**F90 compiler**
SGI Origin 3000 (Silicon Graphics)	IRIX	SGI
IBM SP	AIX	IBM
Compaq True 64 Alpha (DEC ALFA)	OSF1	Compaq
Sun SPARC	Solaris	Sun
PA-RISC (HP 9000 series 700/800)	HP-UX v11	HP
IBM Power6 (pSeries 575)	Linux	IBM
Intel Pentium (32-bit) PC	Linux	GNU (g95)
Intel Pentium (32-bit) PC	Linux	GNU (gfortran)
Intel Pentium (32-bit) PC	Linux	Intel
Intel Pentium (64-bit) PC	Linux	Intel
Intel Itanium (64-bit) PC	Linux	Intel
Intel Pentium (64-bit) PC	Linux	Portland Group
Intel Pentium (32-bit) PC	Linux	Lahey

Intel Pentium (32-bit) PC	MS Windows	Intel
Intel Pentium (64-bit) PC	MS Windows	Intel
Intel Pentium (32-bit) PC	MS Windows	Compaq Visual
Power Mac G4	Mac OS X	IBM

If your computer and available compiler is mentioned in the table, you may consult Section 3.1 for a quick installation of SWAN. Otherwise, read Section 3.2 for a detailed description of the manual installation of SWAN.

Note that for a successful installation, a Perl package must be available on your computer. In most cases, it is available for Linux and a UNIX operating system. Check it by typing `perl -v`. Otherwise, you can download a free distribution for Windows called ActivePerl; see *http://www.activestate.com/activeperl/downloads*. The Perl version should be at least 5.0.0 or higher!

Before installation, the user may first decide how to run the SWAN program. There are three possibilities:

- serial runs,
- parallel runs on shared memory systems or
- parallel runs on distributed memory machines.

For stationary and small-scale computations, it may be sufficient to choose the serial mode, i.e. one SWAN program running on one processor. However, for relatively large CPU-demanding calculations (e.g., instationary or nesting ones), two ways of parallelism for reducing the turn-around time are available:

- The SWAN code contains a number of so-called OpenMP directives that enables the compiler to generate multithreaded code on a shared memory computer. For this, you need a Fortran 90 compiler supporting OpenMP 2.0. The performance is good for a restricted number of threads (< 8). This type of parallelism can be used e.g., on (symmetric) multiprocessors and PC's with dual core or quad core processors.
- If the user want to run SWAN on a relative large number of processors, a message passing model is a good alternative. It is based on independent processors which do not share any memory but are connected via an interconnection network (e.g. cluster of Linux PC's connected via fast Ethernet switches). The implementation is based on the Message Passing Interface (MPI) standard (e.g., MPICH2 distribution, freely available for several platforms, such as Linux and Windows, at *http://www-unix.mcs.anl.gov/mpi/mpich*). The SWAN code contains a set of generic subroutines

that call a number of MPI routines, meant for local data exchange, gathering data, global reductions, etc. This technique is beneficial for larger simulations only, such that the communication times are relatively small compared to the computing times.
- For a proper installation of MPICH2 on Windows (XP/Vista/7), please consult Section 3.4.
- Consult Chapter 6 for parallel, unstructured mesh simulations on distributed memory machines.

Subsections

- Quick installation
- Manual installation
  - Modifications in the source code
  - Compiling and linking SWAN source code

- Building with netCDF support
- Installation of MPICH2 on Windows XP/Vista/7
- Make SWAN documentation

Quick installation

Carry out the following steps for setting up SWAN on your computer.

1. An include file containing some machine-dependent macros must be created first. This file is called `macros.inc` and can be created by typing
2. `make config`
3. Now, SWAN can be built for serial or parallel mode, as follows:

mode	instruction
serial	`make ser`
parallel, shared	`make omp`
parallel, distributed	`make mpi`

4.

IMPORTANT NOTES:

- To Windows users:
  - To execute the above instructions, just open a command prompt.
  - To build SWAN on Windows platforms by means of a Makefile you need a make program. However, such a program is usually not available in MS Windows environment. There are two alternatives to get a make program:
    1. You can download the Visual Studio 2005/2008/2010 Express for free; *http://www.microsoft.com/express*. This contains tools for building applications. It also contains a make program called `NMAKE.EXE`.
    2. You can download `Nmake` 1.5 for Win32 that is freely available (see *http://support.microsoft.com/default.aspx?scid=kb;en-us;Q132084*).
       Run the downloaded file (`Nmake15.exe`) to extract it. Copy both the `NMAKE.EXE` and the `NMAKE.ERR` file to SWAN directory.
  - This setup does support OpenMP for Windows dual core systems, if Intel Fortran compiler is provided.
  - This installation currently supports MPICH2 for Windows XP/Vista/7; older versions are probably not supported. See Section 3.4 for further information.
  - It is assumed that both the directories `include` and `lib` are resided in `C: \PROGRAM FILES \MPICH2`. If not, the file `macros.inc` should be adapted such that they can be found by the Makefile.
- One of the commands `make ser`, `make omp` and `make mpi` must be preceded once by `make config`.

- If desirable, you may clean up the generated object files and modules by typing `make clean`. If you want to delete any stray files from a previous compilation, just type `make clobber`.
- If you are unable to install SWAN using the Makefile and Perl scripts for whatever reason, see Section 3.2 for instructions on manual installation.

Manual installation

Subsections

- Modifications in the source code
- Compiling and linking SWAN source code

Modifications in the source code

To compile SWAN on your computer system properly, some subroutines should be adapted first depending on the operating system, use of compilers and the wish to use MPI for parallel runs. This can be done by removing the switches started with '!' followed by an indentifiable prefix in the first 3 or 4 columns of the subroutine. A Perl script called `switch.pl` is provided in the material that enables the user to quickly select the switches to be removed. This script can be used as follows:

```
perl switch.pl [-dos] [-unix] [-f95] [-mpi] [-pun] [-cray] [-sgi]
               [-cvis] [-timg] [-matl4] [-impi] [-adcirc] [-netcdf] *.ftn
```

where the options are all optionally. The meaning of these options are as follows.

`-dos`, `-unix`

Depending on the operating system, both the TAB and directory separator character must have a proper value (see also Chapter 4). This can be done by removing the switch !DOS or !UNIX, for Windows and UNIX/Linux platforms, respectively, in the subroutines `OCPINI` (in `ocpids.ftn`) and `TXPBLA` (in `swanser.ftn`). For other operating system (e.g., Macintosh), you should change the values of the following variables manually: DIRCH1, DIRCH2 (in `OCPINI`), TABC (in `OCPINI`) and ITABVL (in `TXPBLA`).

`-f95`

If you have a Fortran 95 compiler or a Fortran 90 compiler that supports Fortran 95 features, it might be useful to activate the `CPU_TIME` statement in the subroutines `SWTSTA` and `SWTSTO` (in `swanser.ftn`) by removing the switch !F95 meant for the detailed timings of several parts of the SWAN calculation. Note that this can be obtained with the command TEST by setting **itest=1** in your command file.

`-mpi`

For the proper use of MPI, you must remove the switch !MPI at several places in the file `swanparll.ftn`, `swancom1.ftn` and `swmod1.ftn`.

`-pun`

To enable to do parallel, unstructured mesh simulation, the switch !PUN must be removed at several places in different files.

`-cray`, `-sgi`

If you use a Cray or SGI Fortran 90 compiler, the subroutines `OCPINI` (in `ocpids.ftn`) and `FOR` (in `ocpmix.ftn`) should be adapted by removing the switch !/Cray or !/SGI since, these compilers cannot read/write lines longer than 256 characters by default. By means

of the option RECL in the OPEN statement sufficiently long lines can be read/write by these compilers.

`-cvis`

The same subroutines `OCPINI` and `FOR` need also to be adapted when the Compaq Visual Fortran compiler is used in case of a parallel MPI run. Windows systems have a well-known problem of the inability of opening a file by multiple SWAN executables. This can be remedied by using the option SHARED in the OPEN statement for shared access. For this, just remove the switch !CVIS.

`-timg`

If the user want to print the timings (both wall-clock and CPU times in seconds) of different processes within SWAN then remove the switch !TIMG. Otherwise, no timings will be keeped up and subsequently printed in the `PRINT` file.

`-matl4`

By default, the created binary Matlab files are of Level 5 MAT-File format and are thus compatible with MATLAB version 5 and up. In this case the switch !MatL5 must be removed. However, some machines do not support a 1-byte unit for the record length (e.g. IBM Power6). At those computers, the binary Matlab files must be formatted of Level 4. In this case the switch !MatL4 must be removed while the switch !MatL5 should not be removed. Level 4 MAT-files are compatible with MATLAB versions 4 and earlier. However, they can be read with the later versions of MATLAB.

`-impi`

Some Fortran compilers do not support `USE MPI` statement and therefore, the module `MPI` in `swmod1.ftn` must be included by removing the switch !/impi.

`-adcirc`

To enable to do coupled ADCIRC+SWAN simulation, the switch !ADC must be removed at several places in different files. However, for a standalone SWAN simulation, the switch !NADC must be removed while the switch !ADC should not be removed.

`-netcdf`

For the proper use of netCDF, you must remove the switch !NCF at several places in different files.

For example, you work on a Linux cluster where MPI has been installed and use the Intel Fortran compiler (that can handle Fortran 95 statements), then type the following:

```
perl switch.pl -unix -f95 -mpi *.ftn *.ftn90
```

Note that due to the option `-unix` the extension `ftn` is automatically changed into `f` and `ftn90` into `f90`.

After the necessary modifications are made as described in the previous section, the source code is ready for compilation. All source code is written in Fortran 90 so you must have a Fortran 90 compiler in order to compile SWAN. The source code cannot be compiled with a Fortran 77 compiler. If you intended to use MPI for parallel runs, you must use the command `mpif90` instead of the original compiler command or using the Integrated Development Environment e.g., for Visual Fortran (see Installation and User's Guide for MPICH2).

The SWAN source code complies with the ANSI Fortran 90 standard, except for a few cases, where the limit of 19 continuation lines is violated. We are currently not aware of any compiler that cannot deal with this violation of the ANSI standard.

When compiling SWAN you should check that the compiler allocates the same amount of memory for all INTEGERS, REAL and LOGICALS. Usually, for these variables 4 bytes are allocated, on supercomputers (vector or parallel), however, this sometimes is 8 bytes. When a compiler allocates 8 bytes for a REAL and 4 bytes for an INTEGER, for example, SWAN will not run correctly.

Furthermore, SWAN can generate binary MATLAB files on request, which are unformatted. Some compilers, e.g. Compaq Visual Fortran and Intel Fortran version 9.x (or higher), measured record length in 4-byte or longword units and as a consequence, these unformatted files cannot be loaded in MATLAB. Hence, in such as case a compiler option is needed to request 1-byte units, e.g. for Compaq Visual Fortran this is `/assume:byterecl` and for Intel Fortran version 9.x (or higher) this is `-assume byterecl`.

The modules (in files `swmod1.ftn`, `swmod2.ftn`, `m_constants.ftn90`, `m_fileio.ftn90`, `serv_xnl4v5.ftn90`, `mod_xnl4v5.ftn90`, `SwanGriddata.ftn90`, `SwanGridobjects.ftn90` and `SwanCompdata.ftn90`) must be compiled first. Several subroutines use these modules. These subroutines need the compiled versions of the modules before they can be compiled. Linking should be done without any options nor using shared libraries (e.g. math or NAG). It is recommended to rename the executable to `swan.exe` after linking.

Referring to the previous example, compilation and linking may be done as follows:

```
mpif90 swmod1.f swmod2.f m_constants.f90 m_fileio.f90 serv_xnl4v5.f90
       mod_xnl4v5.f90 SwanGriddata.f90 SwanGridobjects.f90
SwanCompdata.f90
       ocp*.f swan*.f Swan*.f90 -o swan.exe
```

Building with netCDF support

SWAN 40.91A contains extensions that provide netCDF output of spectra and maps of several wave parameters. Carry out the following steps.

1. Make sure that netCDF 4.1.x is compiled and available on your system. For details, consult *http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-install*. The Fortran interface must have been enabled when netCDF was built.
2. Define the variable `NETCDFROOT` in file `macros.inc` to point to the netCDF root directory (e.g. `NETCDFROOT=/usr/local/netcdf-4.1.3` in case of Linux or `NETCDFROOT=C: \PROGRAM FILES \netcdf` in case of Windows). This enables to compile netCDF I/O into SWAN.
3. Build SWAN as usual (e.g. `make mpi`); see Section 3.1.

Installation of MPICH2 on Windows XP/Vista/7

SWAN can be built with support for MPI. It is assumed that MPI has been installed already in the Linux environment. However, this is probably not the case for Windows. This section instructs you how to install MPI in the Windows environment. It is assumed that your OS is either Windows XP, Vista or 7.

- Install MPICH2 on your machine using the installer downloaded from the downloads page *http://www.mcs.anl.gov/research/projects/mpich2/downloads/index.php?s=downloads*.
- During installation select MPICH2 to be installed for `Everyone`.
- Also make sure that you keep the default setting for passphrase (usually `behappy`) during the installation.
- After installing MPICH2 successfully, you may add rules to your firewall to allow the programs `mpiexec.exe` and `smpd.exe` to communicate through firewall. This process depends on your firewall.
- You need to add the `bin` folder of MPICH2 (usually `C: \PROGRAM FILES \MPICH2 \bin`) to your `PATH`.
- Next, open an admin command prompt by right-clicking on the command prompt icon and selecting `run as administrator`. In the command prompt type the following commands in sequence:
  `smpd -install`
  `mpiexec -remove`
  `mpiexec -register`
  `mpiexec -validate` (it should return `SUCCESS`)
  `smpd -status` (it should return 'smpd running on `<hostname>`')
  If the last two commands are successfully, then you are ready to run SWAN in parallel. See Chapter 5.

Note that when you register, you should indicate your username and password as a user. Moreover, if your machine is not part of a domain, do not specify any domain name.

Make SWAN documentation

SWAN comes with 4 detailed documents which are provided as downloadable PDF files as well as browsable web pages:

- The User Manual describes the complete input and usage of the SWAN package.
- The Implementation Manual explains the installation procedure of SWAN on a single- or multi-processor machine with shared or distributed memory.
- The Programming rules is meant for programmers who want to develop SWAN.
- The Technical documentation discusses the mathematical details and the discretizations that are used in the SWAN program.

These documents are written in LaTeX format. If you are new to LaTeX, we recommend to read first the manual LaTeX for dummies that is available in the SWAN material.

The PDF files are very easy to generate by just typing

User dependent changes and the file `swaninit`

SWAN allows you to customize the input and the output to the wishes of your department, company or institute. This can be done by changing the settings in the initialisation file `swaninit`, which is created during the first time SWAN is executed on your computer system. The changes in `swaninit` only affect the runs executed in the directory that contains that file.

A typical initialisation file `swaninit` may look like:

```
    4                               version of initialisation file
Delft University of Technology      name of institute
    3                               command file ref. number
INPUT                               command file name
    4                               print file ref. number
PRINT                               print file name
    4                               test file ref. number
                                    test file name
    6                               screen ref. number
   99                               highest file ref. number
$                                   comment identifier
[TAB]                               TAB character
\                                   dir sep char in input file
/                                   dir sep char replacing previous one
    1                               default time coding option
  100                               speed of processor   1
  100                               speed of processor   2
  100                               speed of processor   3
  100                               speed of processor   4
```

Explanation:

- The version number of the initialisation file is included in the file so that SWAN can verify whether the file it reads is a valid initialisation file. The current version is 4.
- The initialisation file provides a character string containing the name of the institute that may carry out the computations or modifying the source code. You may assign it to the name of your institute instead of 'DELFT UNIVERSITY OF TECHNOLOGY', which is the present value.
- The standard input file and standard print file are usually named `INPUT` and `PRINT`, respectively. You may rename these files, if appropriate.
- The unit reference numbers for the input and print files are set to 3 and 4, respectively. If necessary, you can change these numbers into the standard input and output unit numbers for your installation. Another unit reference number is foreseen for output to screen and it set to 6. There is also a unit number for a separate test print file. In the version that you downloaded from our web page, this is equal to that of the print file so that test output will appear on the same file as the standard print output.

- The comment identifier to be used in the command file is usually '$', but on some computer system this may be inappropriate because a line beginning with '$' is interpreted as a command for the corresponding operating system (e.g., VAX systems). If necessary, change to '!'.
- To insert [TAB] in the initialisation file, just use the TAB key on your keyboard.
- Depending on the operating system, the first directory separation character in `swaninit`, as used in the input file, may be replaced by the second one, if appropriate.
- Date and time can be read and written according to various options. The following options are available:
    1. 19870530.153000 (ISO-notation)
    2. 30-May-87 15:30:00
    3. 05/30/87 15:30:00
    4. 15:30:00
    5. 87/05/30 15:30:00
    6. 8705301530 (WAM-equivalence)

  Note that the ISO-notation has no millenium problem, therefore the ISO-notation is recommended. In case of other options, the range of valid dates is in between January 1, 1911 and December 31, 2010 (both inclusive).

- In case of a parallel MPI run at the machine having a number of independent processors, it is important to assign subdomains representing appropriate amounts of work to each processor. Usually, this refers to an equal number of grid points per subdomain. However, if the computer has processors which are not all equally fast (a so-called heterogeneous machine), then the sizes of the subdomains depend on the speed of the processors. Faster processors should deal with more grid points than slower ones. Therefore, if necessary, a list of non-default processor speeds is provided. The given speeds are in % of default = 100%. As an illustrating example, we have two PC's connected via an Ethernet switch of which the first one is 1.5 times faster than the second one. The list would be
- ```
  150    speed of processor 1
  ```
- ```
  100    speed of processor 2
  ```

  Based on this list, SWAN will automatically distribute the total number of active grid points over two subdomains in an appropriate manner. Referring to the above example, with 1000 active points, the first and second subdomains will contain 600 and 400 grid points, respectively.

Usage of SWAN executable

To help you in editing an command file for SWAN input, the file `swan.edt` is provided.

Two run procedures are provided among the source code, one for the Windows platform, called `swanrun.bat`, and one for the UNIX/Linux platform, called `swanrun`. Basically, the run procedure carries out the following actions:

- Copy the command file with extension `swn` to `INPUT` (assuming `INPUT` is the standard file name for command input, see Chapter 4).
- Run SWAN.
- Copy the file `PRINT` (assuming `PRINT` is the standard file name for print output, see Chapter 4) to a file which name equals the command file with extension `prt`.

On other operating system a similar procedure can be followed. For parallel MPI runs, the program `mpirun` or `mpiexec` is needed and is provided in the MPICH2 distribution.

Before calling the run procedure, the environment variable `PATH` need to be adapted by including the pathname of the directory where `swan.exe` can be found. In case of Windows, this pathname can be specified through the category *System* of *Control Panel* (on the *Advanced* tab, click *Environment Variables*). In case of UNIX or Linux running the bash shell (sh or ksh), the environment variable `PATH` may be changed as follows:

```
export PATH=${PATH}:/usr/local/swan
```

if `/usr/local/swan` is the directory where the executable `swan.exe` is resided. In case of the C shell (csh), use the following command:

```
setenv PATH ${PATH}:/usr/local/swan
```

If appropriate, you also need to add the directory path where the `bin` directory of MPICH2 is resided to `PATH` to have access to the command `mpiexec`.

You may also specify the number of threads to be used during execution of the multithreaded implementation of SWAN on multiprocessor systems. The environment variable for this is `OMP_NUM_THREADS` and can be set like

```
export OMP_NUM_THREADS=4
```

or

```
setenv OMP_NUM_THREADS 4
```

or, in case of Windows,

```
OMP_NUM_THREADS = 4
```

When dynamic adjustment of the number of threads is enabled, the value given in `OMP_NUM_THREADS` represents the maximum number of threads allowed.

The provided run procedures enable the user to properly and easily run SWAN both serial as well as parallel (MPI or OpenMP). Note that for parallel MPI runs, the executable `swan.exe` should be accessible by copying it to all the multiple machines or by placing it in a shared directory. When running the SWAN program, the user must specify the name of the command file. However, it is assumed that the extension of this file is `swn`. Note that contrary to UNIX/Linux, Windows does not distinguish between lowercase and uppercase characters in filenames. Next, the user may also indicate whether the run is serial or parallel. In case of Windows, use the run procedure `swanrun.bat` from a command prompt:

```
swanrun filename [nprocs]
```

where `filename` is the name of your command file without extension (assuming it is `swn`) and `nprocs` indicates how many processes need to be launched for a parallel MPI run (do not type the brackets; they just indicate that `nprocs` is optional). By default, *nprocs* = 1. You may also run on a dual/quad core computer; do not set `nprocs`.

The command line for the UNIX script `swanrun` is as follows:

```
./swanrun -input filename [-omp n | -mpi n]
```

where `filename` is the name of your command file without extension. Note that the script `swanrun` need to be made executable first, as follows:

```
chmod +rx ./swanrun
```

The parameter `-omp n` specifies a parallel run on *n* cores using OpenMP. Note that the UNIX script will set `OMP_NUM_THREADS` to *n*. The parameter `-mpi n` specifies a parallel run on *n* processors using MPI. The parameter `-input` is obliged, whereas the parameters `-omp n` and `-mpi n` can be omitted (default: *n* = 1). To redirect screen output to a file, use the sign >. Use an ampersand to run SWAN in the background. An example:

```
./swanrun -input f31har01 -omp 4 > swanout &
```

For a parallel MPI run, you may also need a `machinefile` that contains the names of the nodes in your parallel environment. Put one node per line in the file. Lines starting with the # character are comment lines. You can specify a number after the node name to indicate how many cores to launch on the node. This is useful e.g., for multi-core processors. The run procedure will cycle through this list until all the requested processes are launched. Example of such a file may look like:

```
# here, eight processes will be launched
node1
node2:2
node4
node7:4
```

Note that for Windows platforms, a space should be used instead of a colon as the separation character in the `machinefile`.

SWAN will generate a number of output files:

- A print file with the name `PRINT` that can be renamed by the user with a batch (DOS) or script (UNIX) file, e.g. with the provided run procedures. For parallel MPI runs, however, a sequence of `PRINT` files will be generated (`PRINT-001`, `PRINT-002`, etc.) depending on the number of processors. The print file(s) contain(s) the echo of the input, information concerning the iteration process, possible errors, timings, etc.
- Numerical output (such as table, spectra and block output) appearing in files with user provided names.
- A file called `Errfile` (or renamed by the run procedures as well as more than one file in case of parallel MPI runs) containing the error messages is created only when SWAN produces error messages. Existence of this file is an indication to study the results with more care.
- A file called `ERRPTS` (or renamed by the run procedures as well as more than one file in case of parallel MPI runs) containing the grid-points, where specific errors occured during the calculation, such as non-convergence of an iterative matrix-solver. Existence of this file is an indication to study the spectrum in that grid-point with more care.

If indicated by the user, a single or multiple hotfiles will be generated depending on the number of processors, i.e. the number of hotfiles equals the number of processors (see the User Manual). Restarting a (parallel MPI) run can be either from a single (concatenated) hotfile or from multiple hotfiles. In the latter case, the number of processors must be equal to the number of generated hotfiles. If appropriate, the single hotfile can be created from a set of multiple hotfiles using the program `hcat.exe` as available from SWAN version 40.51A. This executable is generated from the Fortran program `swanhcat.ftn`. A self-contained input file `hcat.nml` is provided. This file contains, e.g. the (basis) name of the hotfile. To concatenate the multiple hotfiles into a single hotfile just execute `hcat.exe`.

Compiling and running unstructured mesh SWAN in parallel

In order to efficiently carry out high-resolution simulations with the unstructured mesh version of SWAN, a parallel code is build using message passing paradigm and tested on a commodity computer cluster. In parallelizing unstructured mesh version of SWAN, we employ the same paradigm as applied in the circulation model ADCIRC (*http://www.adcirc.org*) and so the parallel version of the unstructured mesh SWAN model uses the same domain decomposition and localized communication as the parallel version of ADCIRC. For a proper use, version **50** of ADCIRC must be employed.

For details on the implementation and use of the coupled ADCIRC + SWAN model, go to *http://www.caseydietrich.com/swanadcirc/*.

It is assumed that the unstructured mesh is stored in the file `fort.14` and the associated model domain information is stored in `fort.15` and that these files are yet available.

Subsections

- Code compilation
- Running the model

Code compilation

The parallel version of the unstructured SWAN model utilizes the parallel infrastructure from parallel ADCIRC. Specifically, it needs the `MKDIR`, `SIZES`, `GLOBAL` and `MESSENGER` object files. So those need to be compiled before the parallel, unstructured mesh SWAN model is compiled. Also, for mesh partition a program called `adcprep` need to be compiled as well. These compilations are performed typically by navigating to the directory `work` and typing

```
make adcprep padcirc
```

Make sure that the `-DADCSWAN` flag has been added to the `DPRE` line in the `cmplrflags.mk` file.

To compile the parallel version of the unstructured mesh SWAN model, you probably first need to remove the files that were created during the compilation of the commonly used SWAN model. You can do this by typing

```
make clobber
```

to remove the stray files. If necessary, type

```
make config
```

to have SWAN attempt to create a file `macros.inc` with compiler flags for your system. Be sure to complete the line

```
O_DIR = ../work/odir4/
```

where the abovementioned object files are resided. Then type

```
make punswan
```

to make the parallel, unstructured mesh version of SWAN. It will follow the SWAN convention of assigning the name `swan.exe` to the executable.

Running the model

The main difference between running commonly used SWAN in parallel and running the unstructured mesh version of SWAN in parallel on a multi-core cluster is the need to explicitly decompose the unstructured mesh and input files, e.g. `fort.14` and `fort.15`, into smaller pieces, so that each piece can run on its own core on the cluster. The program `adcprep` is used to perform the decomposition. The actual mesh partitioning is done by the well-known METIS package (*http://glaros.dtc.umn.edu/gkhome/views/metis*).

In order to break up the original input files (which describe the full domain) into smaller pieces (called subdomains), go to the directory where the input files are located and execute `adcprep`. You need to specify the number of cores that you plan to use for the simulation. The program will decompose your input files into that number of subdomains and copy each of them into the corresponding local PE sub-directories. Finally, it will copy the SWAN command file into the local PE sub-directories. If you decide later that you want to run on more (or fewer) cores, you must perform this step again.

So, more specifically, run `adcprep` and indicate the number of processors. First, you need to do mesh partition using METIS by selecting

```
1. partmesh
```

Next, rerun `adcprep` and continue with full pre-processing by selecting

```
2. prepall
```

or

```
3. prepspec
```

and specify the necessary ADCIRC files (`fort.14` and `fort.15`; you may skip the file `fort.19`). Be sure that the initialisation file `swaninit` has been created before running `adcprep`. You may adapt this initialisation file by renaming `INPUT` to your own SWAN command file. Otherwise, you may copy your command file to `INPUT`.

After the mesh and input files have been prepared with `adcprep`, the actual SWAN calculation is performed using `swan.exe`. This program performs the calculation on each subdomain and uses MPI to provide ongoing communication between the subdomains. The command required for actually starting the SWAN job is highly dependent on the cluster software and queueing system. Either way, use the new executable to run this model in much the same way you would run the parallel SWAN; see Chapter 5.

After the simulation is finished the output data will be automatically merged. With respect to the generation of hotfiles, it is usually done in different PE sub-directories. So, in principle, in case of restarting, the same number of processors must be employed. Instead, a

single hotfile can be created from the existing local PE hotfiles using the program `unhcat.exe`, as available from SWAN version 40.91. This executable is generated from the Fortran program `HottifySWAN.ftn90`. To concatenate the multiple hotfiles into a single hotfile (*globalization*) just execute `unhcat.exe`. With this program, it is also possible to create a set of local PE hotfiles from an existing global hotfile (*localization*) over a different number of processors.

Testing the system

The SWAN system consists of one executable file (`swan.exe`), a command file (`swan.edt`) and a run procedure (`swanrun.bat` or `swanrun`). Depending on your system, you may use 32-bit or 64-bit executable. These executables for Windows XP/Vista/7 can be obtained from the SWAN web site. The input and output to a number of test problems is provided on the SWAN web page. The files with extension `swn` are the command files for these tests; the files with extension `bot` are the bottom files for these tests, etc. This input can be used to make a configuration test of SWAN on your computer. Compare the results with those in the provided output files. Note that the results need not to be identical up to the last digit.

To run the SWAN program for the test cases, at least 50 MBytes of free internal memory is recommended. For more realistic cases 100 to 500 MBytes may be needed, whereas for more simple stationary or 1D cases significant less memory is needed (less than 5 MBytes for 1D cases).